

Jeffreys Bayes Factor Approximation

The approximation

Jeffreys (1961), p. 246 ff.:

Our problem is to compare a specially suggested value of a new parameter, often 0, with the aggregate of other possible values. We do this by enunciating the hypothesis q that the parameter has the suggested value, and q' , that it has some other value to be determined from the observations. We shall call q the null hypothesis, following Fisher, and q' the alternative hypothesis. To say that we have no information initially as to whether the new parameter is needed or not we must take

$$P(q | H) = P(q' | H) = \frac{1}{2}$$

But q' involves an adjustable parameter, α say, and

$$P(q' | H) = \sum P(q', \alpha | H)$$

over all possible values of α . We take α to be zero on q . [...] If the maximum likelihood solution for α is $a \pm s$, the chance of finding a in a particular range, given q is nearly

$$P(da | qH) = \frac{1}{\sqrt{(2\pi)s}} \exp(-\frac{a^2}{2s^2}) da$$

and the chance, given q' and a particular value of α , is

$$P(da | q'\alpha H) = \frac{1}{\sqrt{(2\pi)s}} f(\alpha) \exp(-\frac{(a - \alpha)^2}{2s^2}) d\alpha$$

Hence, by the principle of inverse probability

$$P(q | aH) \propto \frac{1}{\sqrt{(2\pi)s}} \exp(-\frac{a^2}{2s^2}), P(q'd\alpha | aH) \propto \frac{1}{\sqrt{(2\pi)s}} f(\alpha) \exp(-\frac{(a - \alpha)^2}{2s^2}) d\alpha$$

We shall in general write

$$K = \frac{P(q | \theta H)}{P(q' | \theta H)} / \frac{P(q | H)}{P(q' | H)}$$

Note: The second term (divisor) can be canceled because it is equal to one if the prior probabilities are equal to one.

[...] If the number of observations, n , is large, s is usually small like $n^{-1/2}$. Then, if $a = 0$ and n large, K will be large of order $n^{1/2}$, since $f(a)$ is independent of n . Then the observations support q , that is, they say that the new parameter is probably not needed. But if $|a|$ is much larger than s the exponential will be small, and the observations will support the need for the new parameter.

In other papers (1977, 1980) Jeffreys therefore expresses K as

$$K = \frac{P(q | \theta H)}{P(q' | \theta H)} = A n^{1/2} \exp\left\{-\frac{(a - \alpha_0)^2}{2s_a^2}\right\},$$

where A is a constant.

Try it in R

The exponential can be defined based on a t-value:

$$t = \frac{a - \alpha_0}{s}$$

```
# Approximate BF10 is 1/K
BFapprox <- function(tval, n){
  1/(sqrt(n)*exp(-0.5*tval^2))
}
```

```
# Compare to BayesFactor
library(BayesFactor)
```

```
## Loading required package: coda
```

```
## Loading required package: Matrix
```

```
## *****
```

```
## Welcome to BayesFactor 0.9.12-4.2. If you have questions, please contact Richard Morey (richarddmorey@ucsd.edu)
```

```
##
```

```
## Type BFManual() to open the manual.
```

```
## *****
```

```
ttest.tstat(t = 1.5, n1 = 50, simple=T)
```

```
##          B10
```

```
## 0.4384737
```

```
BFapprox(tval=1.5, n = 50)
```

```
## [1] 0.4356084
```

```
tval <- seq(-5, 5, length.out=1001)
```

```
Ns <- seq(10, 500, by=10)
```

```
cond <- expand.grid(tval, Ns)
```

```
cond$BFapprox <- apply(cond[, 1:2], 1, FUN = function(x) BFapprox(t=x[1], n=x[2]))
```

```
cond$BFttest <- apply(cond[, 1:2], 1, FUN = function(x) unname(ttest.tstat(t=x[1], n1=x[2], simple=T)))
```

```
colnames(cond)[1:2] <- c("tval", "N")
```

Check Quality of Approximation with Graphs

Colors:

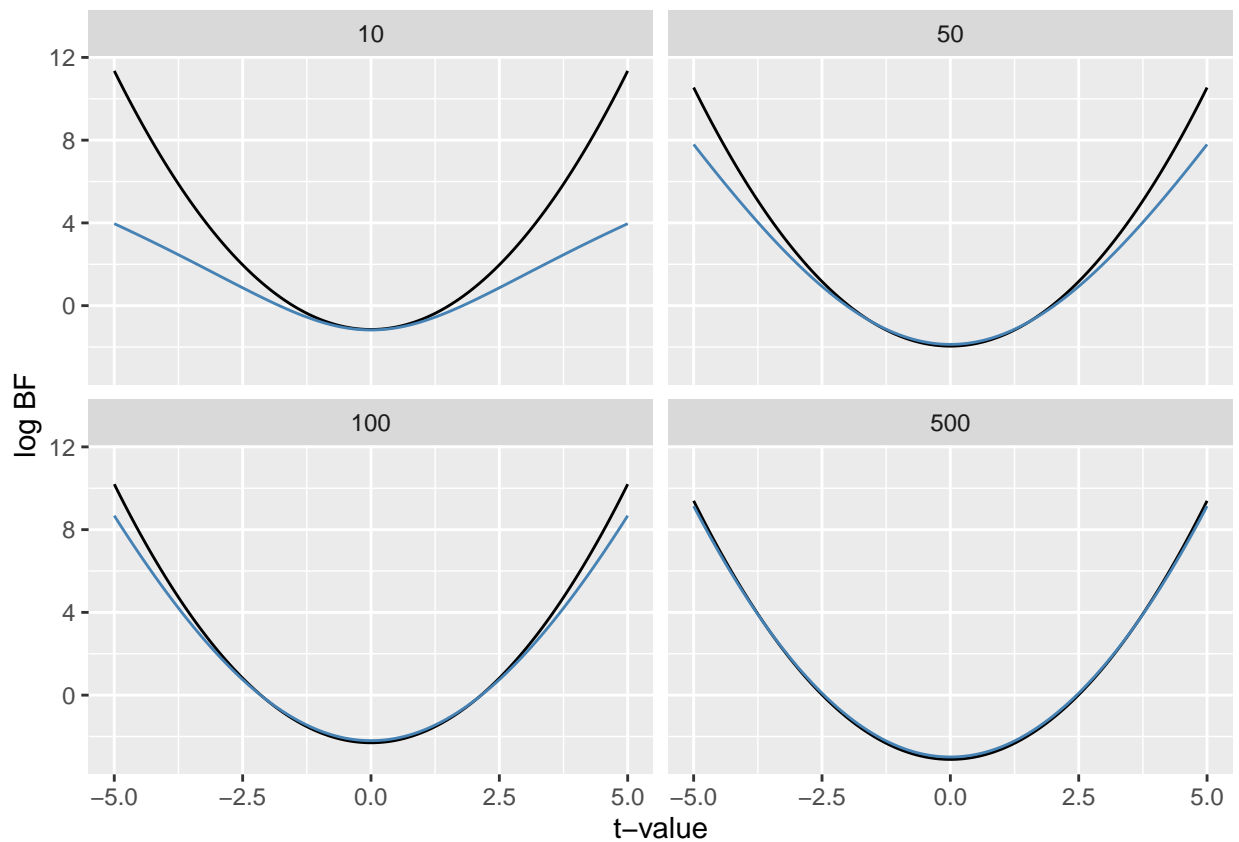
- Black: Jeffreys' approximation
- Blue: Bayes factor

```
library(dplyr)
```

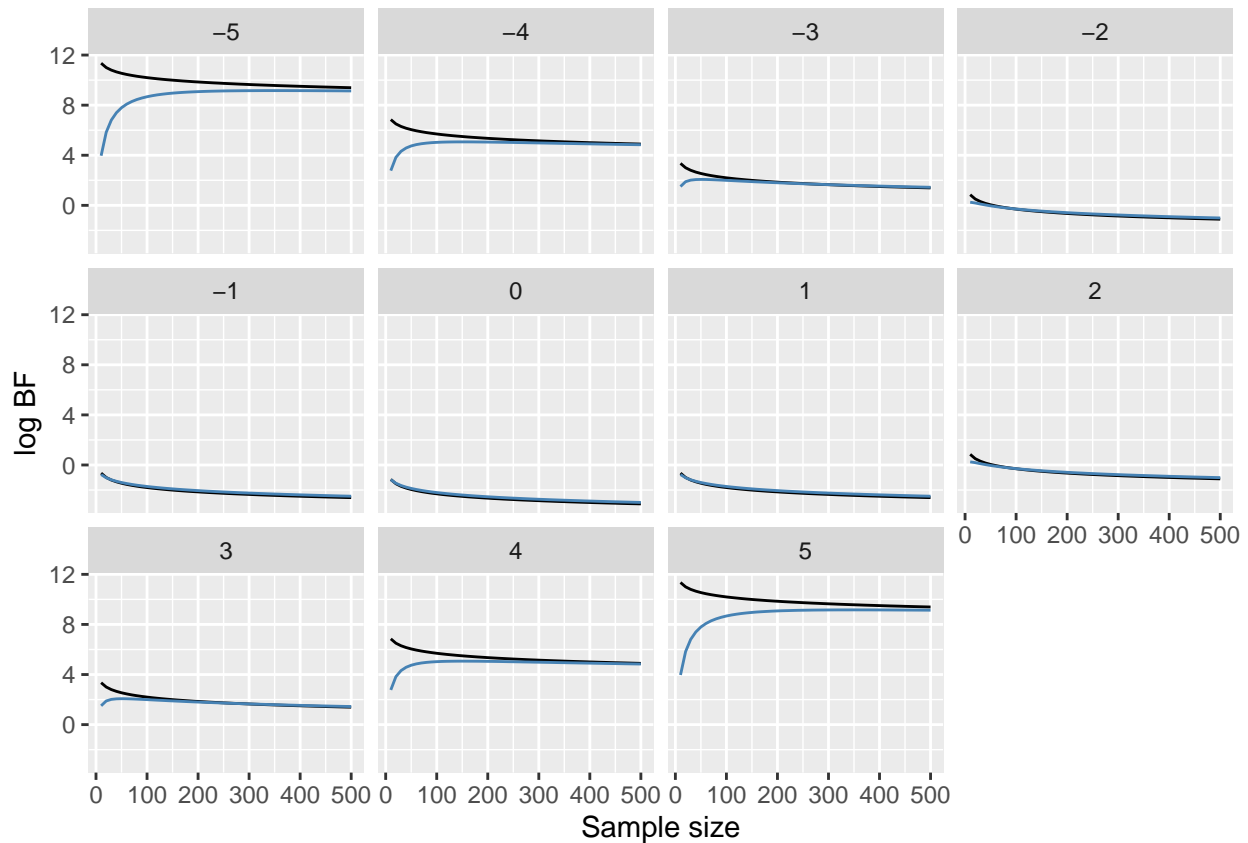
```
##  
## Attaching package: 'dplyr'  
## The following objects are masked from 'package:stats':  
##  
##   filter, lag  
## The following objects are masked from 'package:base':  
##  
##   intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

```
cond %>% filter(N %in% c(10, 50, 100, 500)) %>%  
  ggplot(aes(tval, log(BFapprox))) +  
  geom_line() +  
  facet_wrap(~ N) +  
  geom_line(aes(tval, log(BFttest)), color = "steelblue") +  
  labs(y = "log BF", x = "t-value")
```



```
cond %>% filter(tval %in% c(-5:5)) %>%  
  ggplot(aes(N, log(BFapprox))) +  
  geom_line() +  
  facet_wrap(~ tval) +  
  geom_line(aes(N, log(BFttest)), color="steelblue") +  
  labs(y = "log BF", x = "Sample size")
```



Observations

- Approximation is worse for large t-values
- Approximation is worse for small sample sizes
- For small sample sizes curvature of $BF \sim \text{sample size}$ is reversed

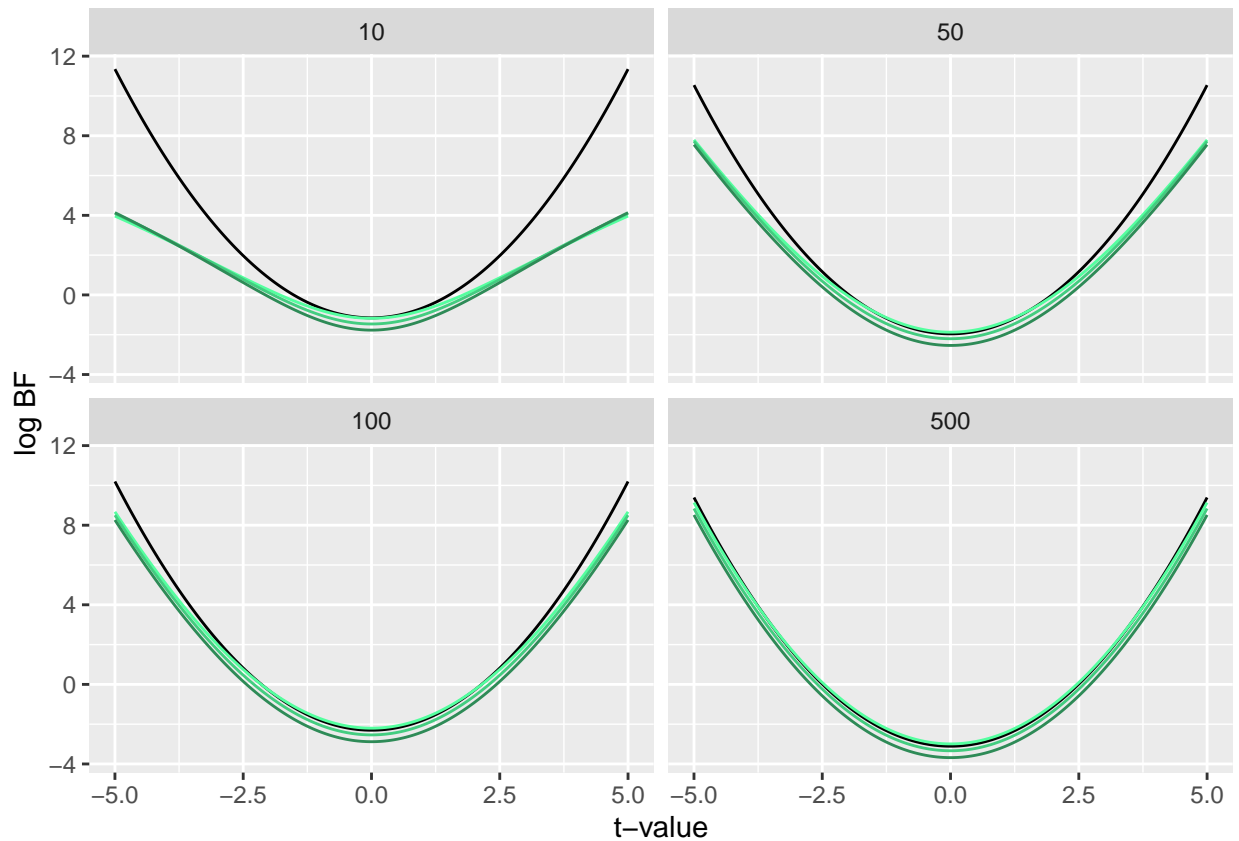
Approximation Quality with Different Priors

Colors:

- Black: Approximation
- Green: Bayes factor (Darker colors used for wider priors. rscale: medium - wide - ultrawide according to *BayesFactor* package)

```
cond$BFwide <- apply(cond[, 1:2], 1, FUN = function(x) unname(ttest.tstat(t=x[1], n1=x[2], simple=T, rscale="medium")))
cond$BFultrawide <- apply(cond[, 1:2], 1, FUN = function(x) unname(ttest.tstat(t=x[1], n1=x[2], simple=T, rscale="ultrawide")))
```

```
cond %>% filter(N %in% c(10, 50, 100, 500)) %>%
  ggplot(aes(tval, log(BFapprox))) +
  geom_line() +
  facet_wrap(~ N) +
  geom_line(aes(tval, log(BFttest)), color = "seagreen1") +
  geom_line(aes(tval, log(BFwide)), color = "seagreen3") +
  geom_line(aes(tval, log(BFultrawide)), color="seagreen") +
  labs(y = "log BF", x = "t-value")
```



Observations

- Same pattern as before
- Differences between priors are most relevant for small t-values, differences BF vs. approximation are clearest for large t-values

```
library(BFDA)
```

```
## Loading required package: doParallel
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
## This is BFDA 0.5.0
```

```
## BFDA is BETA software! Please report any bugs.
```

```
##
```

```
## Attaching package: 'BFDA'
```

```
## The following object is masked from 'package:stats':
```

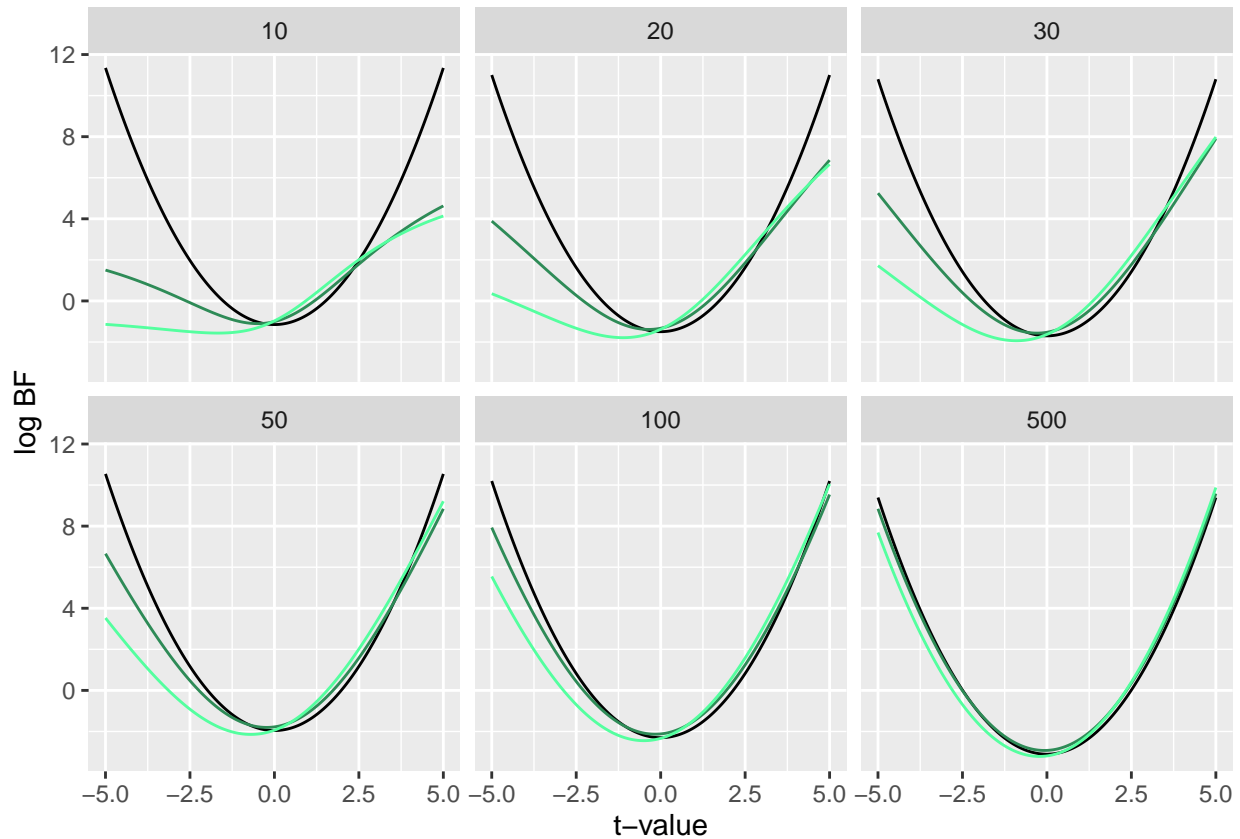
```
##
```

```
## SSD
```

```
cond$BFinf1 <- apply(cond[, 1:2], 1, FUN = function(x) BFDA::bf10_normal(t=x[1], n1=x[2], independentS
cond$BFinf2 <- apply(cond[, 1:2], 1, FUN = function(x) BFDA::bf10_normal(t=x[1], n1=x[2], independentS
```

Graph: Darker color used for wider prior

```
cond %>% filter(N %in% c(10, 20, 30, 50, 100, 500)) %>%
  ggplot(aes(tval, log(BFapprox))) +
  geom_line() +
  facet_wrap(~ N) +
  geom_line(aes(tval, log(BFinf1)), color = "seagreen") +
  geom_line(aes(tval, log(BFinf2)), color = "seagreen1") +
  labs(y = "log BF", x = "t-value")
```



Observations

- With informed priors, the approximation breaks - especially for small sample sizes
- Evidence is no longer symmetric
- It looks like there is no easy way to make the approximation more accurate (e.g., change constant A), at least for small sample sizes

Improve Approximation Quality for Default Prior

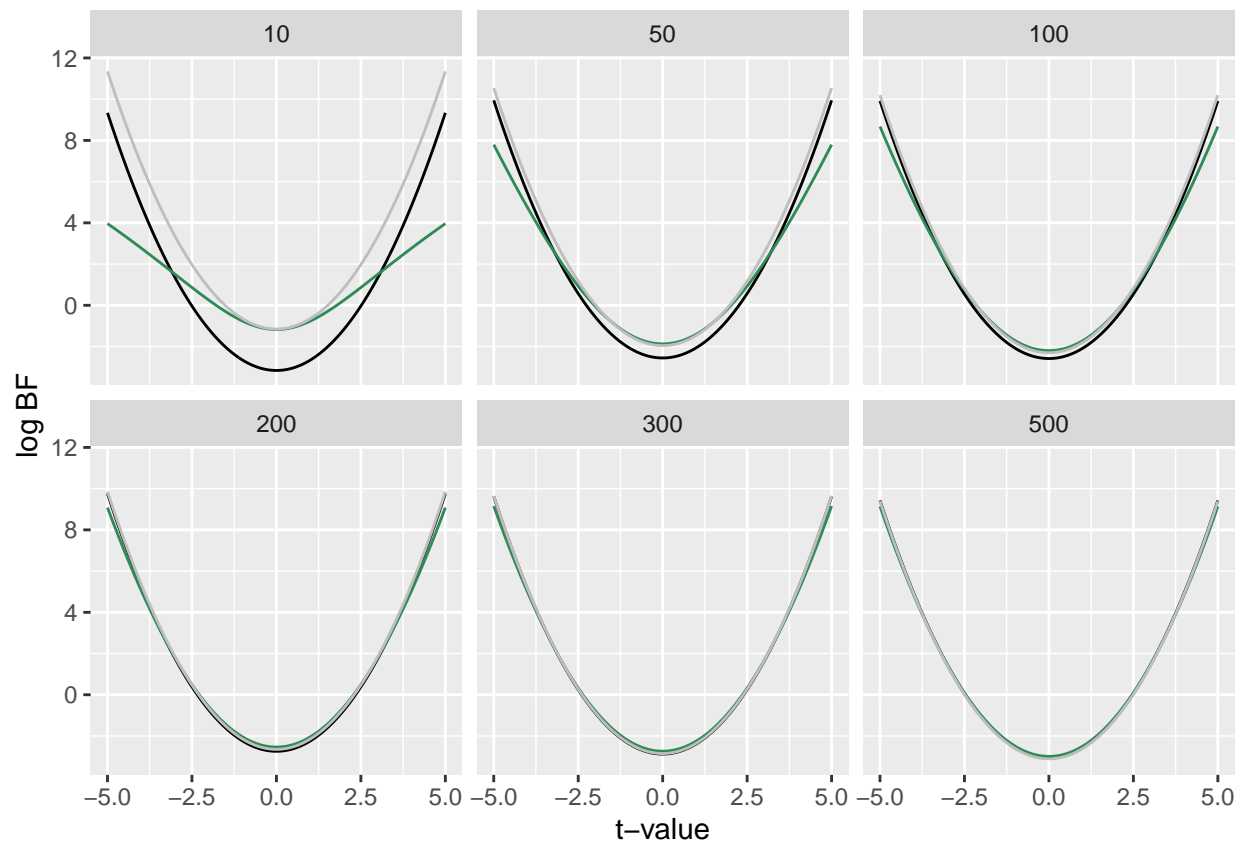
Graph: Gray line = approximate BF for $A = 1$, black line = approximate BF for optimized A , green line = default BF

```
BFapproxA <- function(A, tval, n){
  1/(A*sqrt(n)*exp(-0.5*tval^2))
}

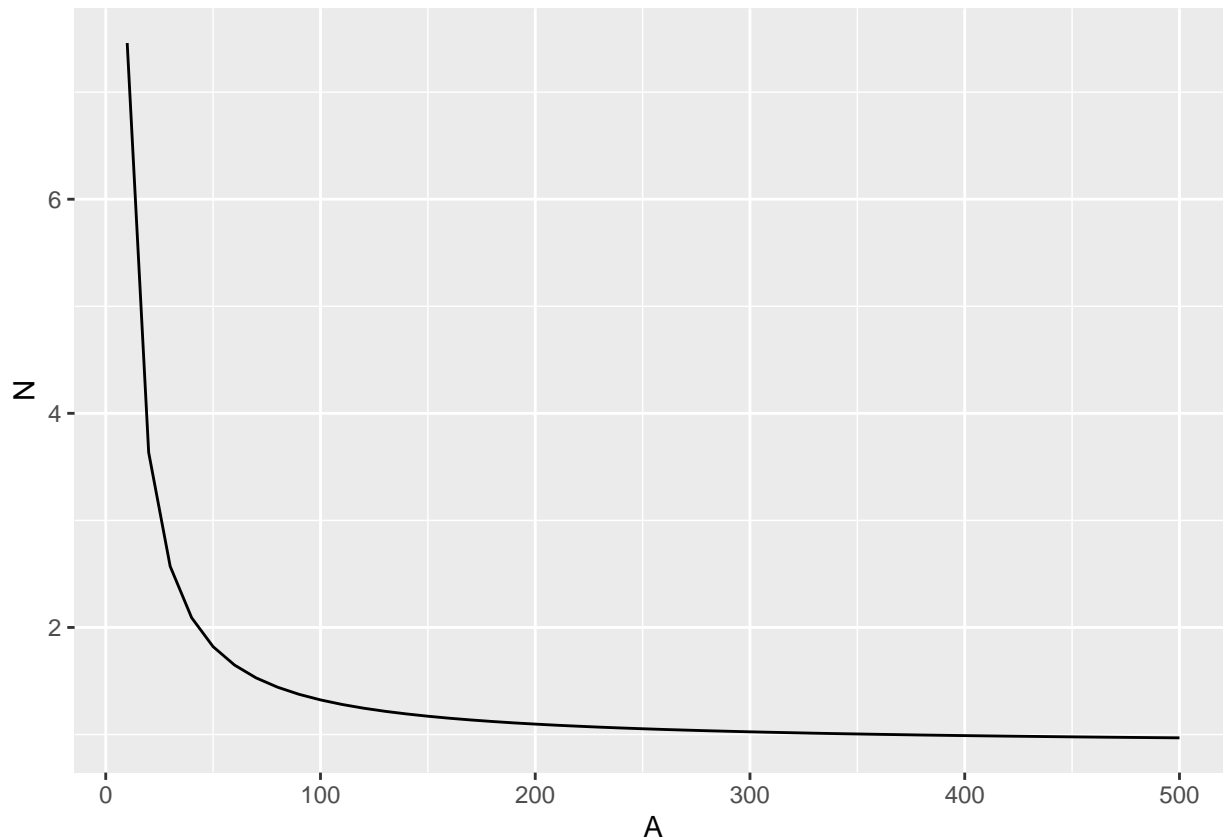
objFun <- function(A, tval, n, BFfs){
  sum((log(BFapproxA(A, tval, n)) - log(BFfs))^2)
}
```


[illegible]


```
geom_line(aes(tval, log(BFttest)), color = "seagreen") +
geom_line(aes(tval, log(BFapprox)), color = "gray") +
labs(y = "log BF", x = "t-value")
```



```
cond %>%
  ggplot(aes(N, As)) +
  geom_line() +
  labs(y = "N", x = "A")
```



Observations:

- If we simply adjust the constant A as Jeffreys suggested, the Bayes factor gets “corrected” towards extreme values of t , i.e., the approximation gets slightly better in the tails, but (a lot) worse for small t -values.
- Since the approximation already worked well for large sample sizes, A is close to 1 for large sample sizes

Polynomial Approximation

For any sample size, the relationship between t -values and log Bayes factors can be described by a parabola shape. This can be approximated using a second-degree polynomial of the form $BF_{10}(t) \approx \beta_0 + \beta_1 t^2$.

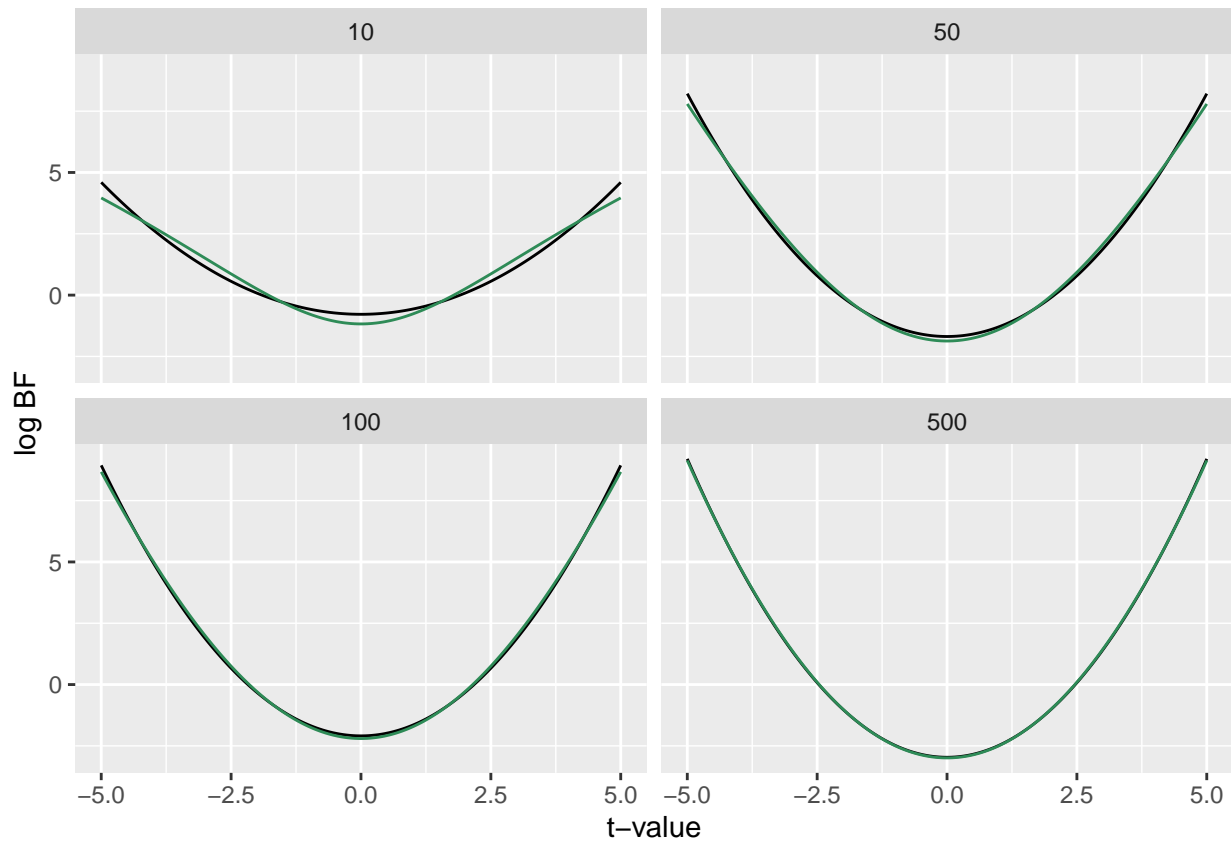
```
fitfun <- function(tval, n, BFs){
  logBFs <- log(BFs)
  lm(logBFs ~ I(tval^2))
}

cond2 <- cond

for(i in Ns){
  fit <- fitfun(tval=tval, n=i, BFs = cond$BFttest[cond$N == i])
  cond2$BFapprox[cond2$N == i] <- exp(unlist(predict(fit)))
}

cond2 %>% filter(N %in% c(10, 50, 100, 500)) %>%
  ggplot(aes(tval, log(BFapprox))) +
  geom_line() +
```

```
facet_wrap(~ N) +
geom_line(aes(tval, log(BFttest)), color = "seagreen") +
labs(y = "log BF", x = "t-value")
```



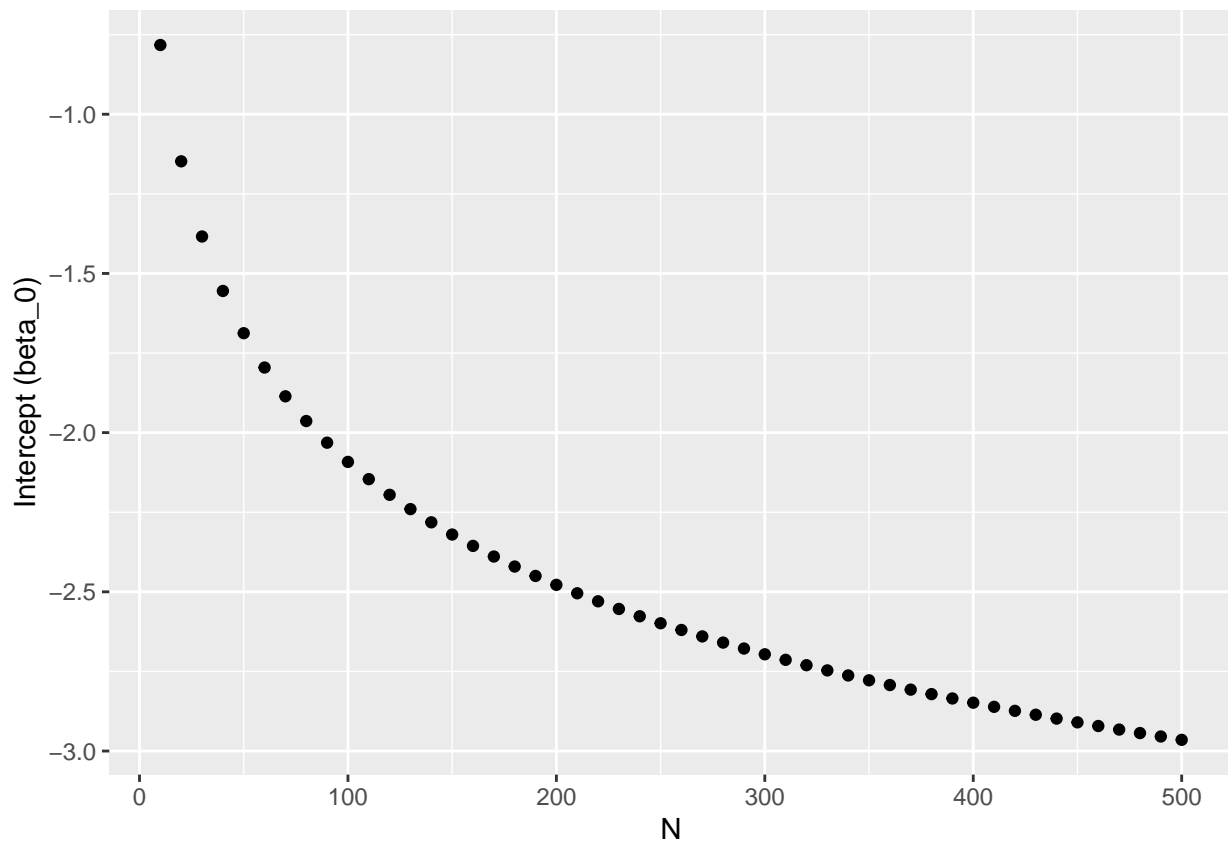
Observations:

- We can obtain reasonably good approximations with a 2nd degree polynomial
- The polynomial approximations also get better with increasing sample size
- They are better for small sample sizes (and extreme t-values) than Jeffreys' approximation

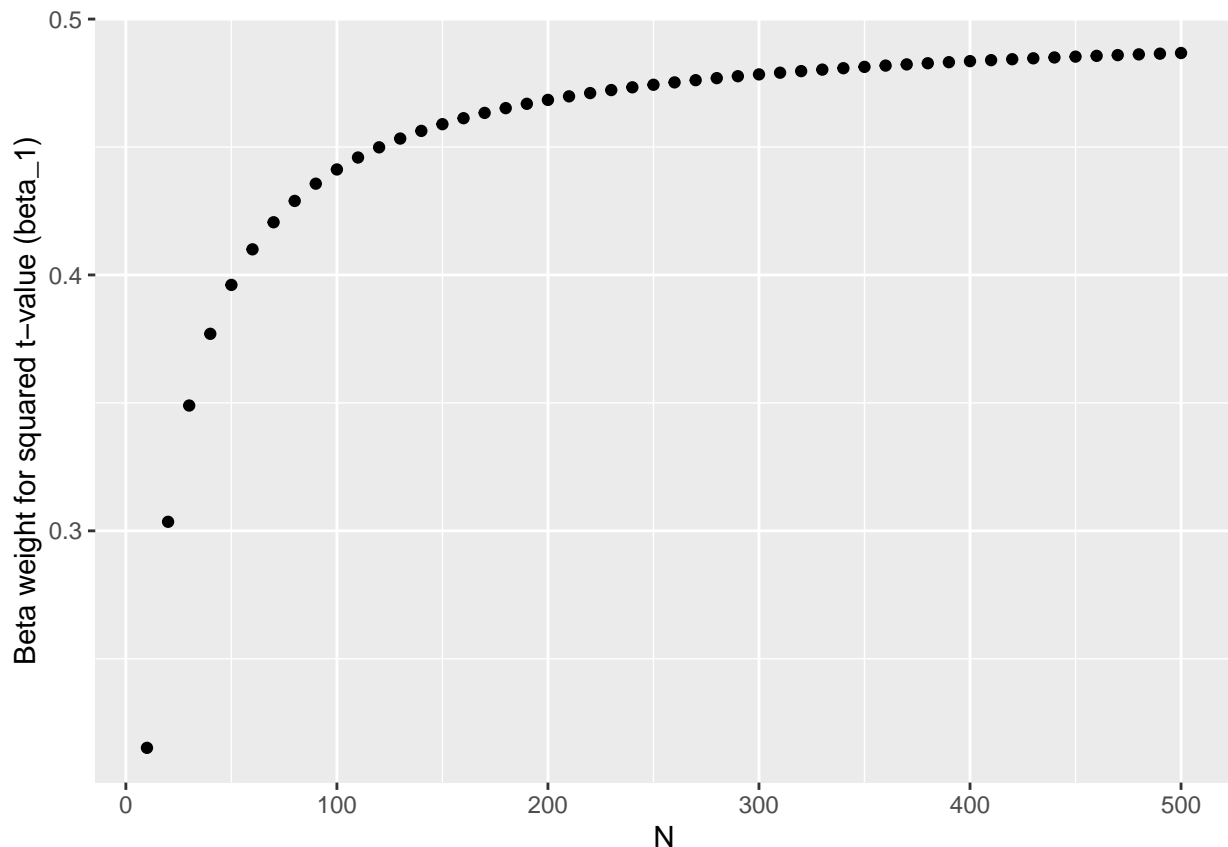
Below, plots of the coefficients:

```
coefs <- matrix(ncol=2)
for(i in Ns){
  fit <- fitfun(tval=tval, n=i, BFts = cond$BFttest[cond$N == i])
  coefs <- rbind(coefs, fit$coefficients)
}
coefData <- as.data.frame(coefs[-1,])
coefData$N <- Ns

coefData %>%
  ggplot(aes(N, `(Intercept)`)) +
  geom_point() +
  labs(y = "Intercept (beta_0)")
```



```
coefData %>%  
  ggplot(aes(N, `I(tval^2)`)) +  
  geom_point() +  
  labs(y = "Beta weight for squared t-value (beta_1)")
```



Observations:

- Clearly, the coefficients depend on the sample size
- The relationship between sample size and the coefficients of the approximating function also looks very systematic

Using Approximations for BFDA

Idea of this section: Conduct a BFDA for a t-test with a default prior with the BayesFactor package or the Bayes factor approximation methods. Check which methods is fastest and (for the approximations) yields results that are most comparable to the “true” BFDA results.

```
delta <- 0.5
N <- 50

BFapproxJef <- function(tval, n){
  1/(sqrt(n)*exp(-0.5*tval^2))
}

## Generate data (t-values) based on effect size
tval <- NA
for(i in 1:10000){
  x <- rnorm(N, delta, sd=1)
  tval[i] <- t.test(x)$statistic
}

res <- as.data.frame(cbind(tval))
```

```
# Calculate BFs using the BayesFactor package
a <- Sys.time()
res$logBF <- log(apply(res, 1, function(x) unname(ttest.tstat(x, n1 = N, rscale = "medium", simple=T))))
print(Sys.time()-a) # this takes roughly 40-45 seconds
```

Time difference of 46.7709 secs

```
# Calculate approximate BFs using Jeffreys's approximation
a <- Sys.time()
res$logBFapproxJeffreys <- log(BFapproxJef(res$tval, n = N))
print(Sys.time()-a) # this takes roughly 0.01 seconds
```

Time difference of 0.001798868 secs

```
# Calculate approximate BFs using a polynomial approximation
a <- Sys.time()
trange <- range(tval) # Find a good BF value approximation function for the t-value range
tfit <- seq(trange[1], trange[2], length.out = 50)
BFs <- apply(cbind(tfit), 1, function(x) unname(ttest.tstat(x, n1=N, rscale="medium", simple=T)))
fit <- fitfun(tfit, n=N, BFs=BFs)
res$logBFapproxLin <- fit$coefficients[1] + fit$coefficients[2]*tval^2
print(Sys.time() - a) # this takes roughly 0.25 seconds
```

Time difference of 0.221024 secs

Log Bayes factor Distribution: Quantiles

```
summary(res$logBF)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.866   1.743   3.515   3.886   5.629  20.129
```

```
summary(res$logBFapproxJeffreys)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.950   2.143   4.388   5.124   7.291  35.835
```

```
summary(res$logBFapproxLin)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.7692  1.7010  3.0557  3.5002  4.8076  22.0336
```

Percentage of Bayes factors crossing thresholds

Lower thresholds

```
thresholds <- log(c(1/30, 1/10, 1/6, 1/3))
apply(cbind(thresholds), 1, function(x) sum(res$logBF<x)/nrow(res))
```

```
## [1] 0.0000 0.0000 0.0005 0.0123
```

```
apply(cbind(thresholds), 1, function(x) sum(res$logBFapproxJeffreys<x)/nrow(res))
```

```
## [1] 0.0000 0.0000 0.0013 0.0127
```

```
apply(cbind(thresholds), 1, function(x) sum(res$logBFapproxLin<x)/nrow(res))
```

```
## [1] 0 0 0 0
```

Upper thresholds


```

thresholds <- log(c(3, 6, 10, 30))
apply(cbind(thresholds), 1, function(x) sum(res$logBF>x)/nrow(res))

## [1] 0.8329 0.7422 0.6688 0.5146

apply(cbind(thresholds), 1, function(x) sum(res$logBFapproxJeffreys>x)/nrow(res))

## [1] 0.8590 0.7889 0.7297 0.6058

apply(cbind(thresholds), 1, function(x) sum(res$logBFapproxLin>x)/nrow(res))

## [1] 0.8546 0.7306 0.6328 0.4409

```

Observations

- Approximations are far from perfect
- Jeffreys approximation tends to overestimate evidence, polynomial approximation tends to underestimate
- Approximation quality likely changes with sample size (as shown before) and effect size
- Jeffreys approximation cannot adapt to different prior distributions, will therefore depend (slightly) on the prior used

Checking for more sample sizes and effect sizes

```

Ns <- c(10, 50, 100, 500)
deltas <- c(0, 0.1, 0.3, 0.5, 0.8)
priorwidth <- c(0.707, 1)

conds <- expand.grid(Ns, deltas)
tvals <- simplify2array(apply(conds, 1, function(x){
  tval <- NA
  for(i in 1:5000){
    samp <- rnorm(x[1], x[2], sd=1)
    tval[i] <- t.test(samp)$statistic
  }
  return(tval)
})))

# BFs prior1
# BFs prior2
# BFs approx. Jeffreys
# BFs approx. polynomial (prior1)
# BFs approx. polynomial (prior2)

res <- array(NA, dim=c(5000, (length(Ns)*length(deltas)), 5))

logBFapproxLin <- function(tval, N, rscale){
  # Find a good BF value approximation function for the t-value range
  trange <- range(tval)
  tfit <- seq(trange[1], trange[2], length.out = 50)
  BFs <- apply(cbind(tfit), 1, function(x) unname(ttest.tstat(x, n1=N, rscale=rscale, simple=T)))
  # Apply approximation
  fit <- fitfun(tfit, n=N, BFs=BFs)
  fit$coefficients[1] + fit$coefficients[2]*tval^2
}

for(i in 1:(length(Ns)*length(deltas))){

```

```
res[, i, 1] <- log(apply(cbind(tvals[,i]), 1, function(x) unname(ttest.tstat(x, n1 = conds[i,1], rscale="medium"))))
res[, i, 2] <- log(apply(cbind(tvals[,i]), 1, function(x) unname(ttest.tstat(x, n1 = conds[i,1], rscale="wide"))))
res[, i, 3] <- log(BFapproxJef(tvals[,i], n = conds[i,1]))
res[, i, 4] <- logBFapproxLin(tvals[,i], N = conds[i, 1], rscale="medium")
res[, i, 5] <- logBFapproxLin(tvals[,i], N = conds[i, 1], rscale="wide")
}
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

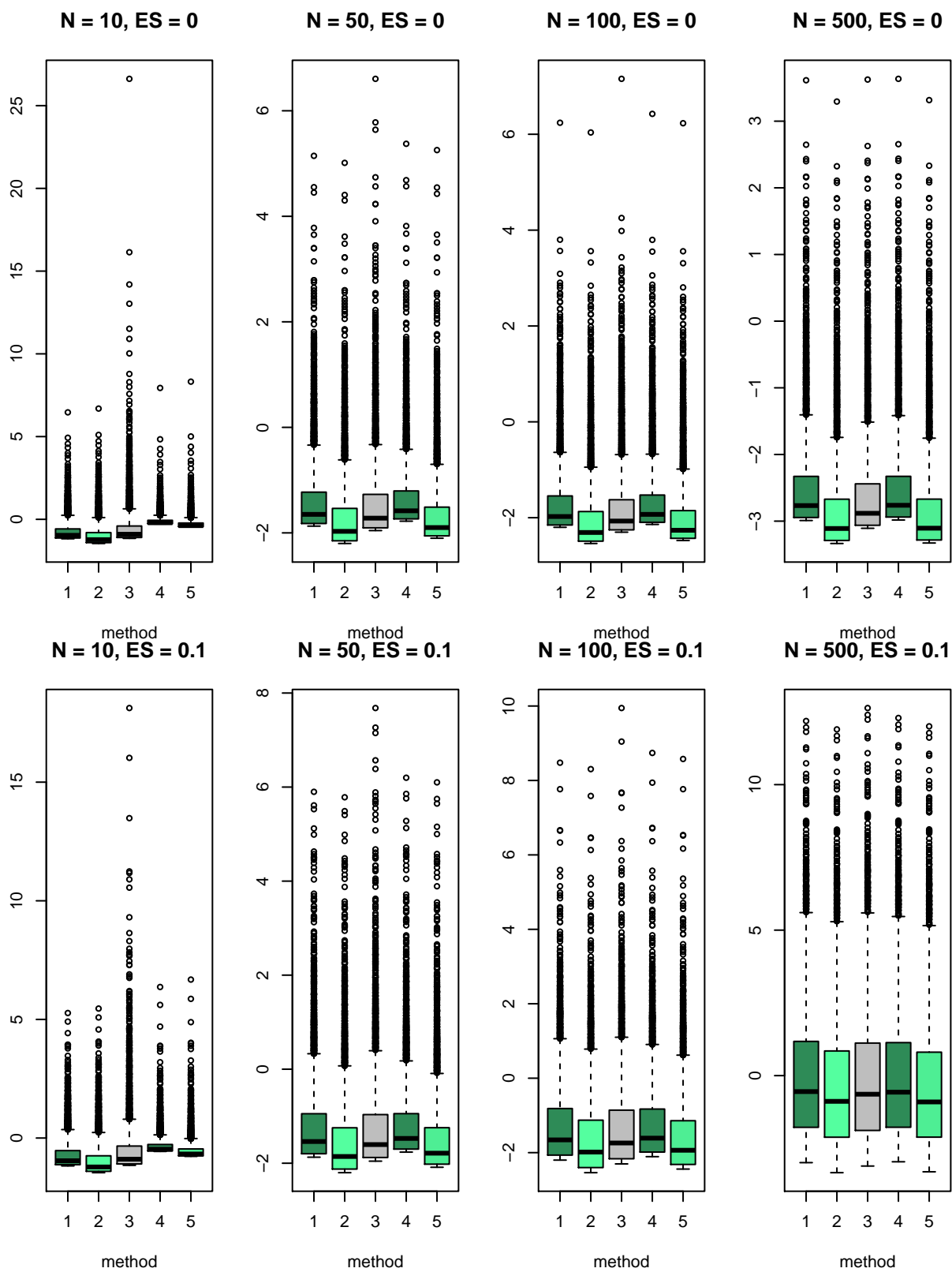
[illegible]

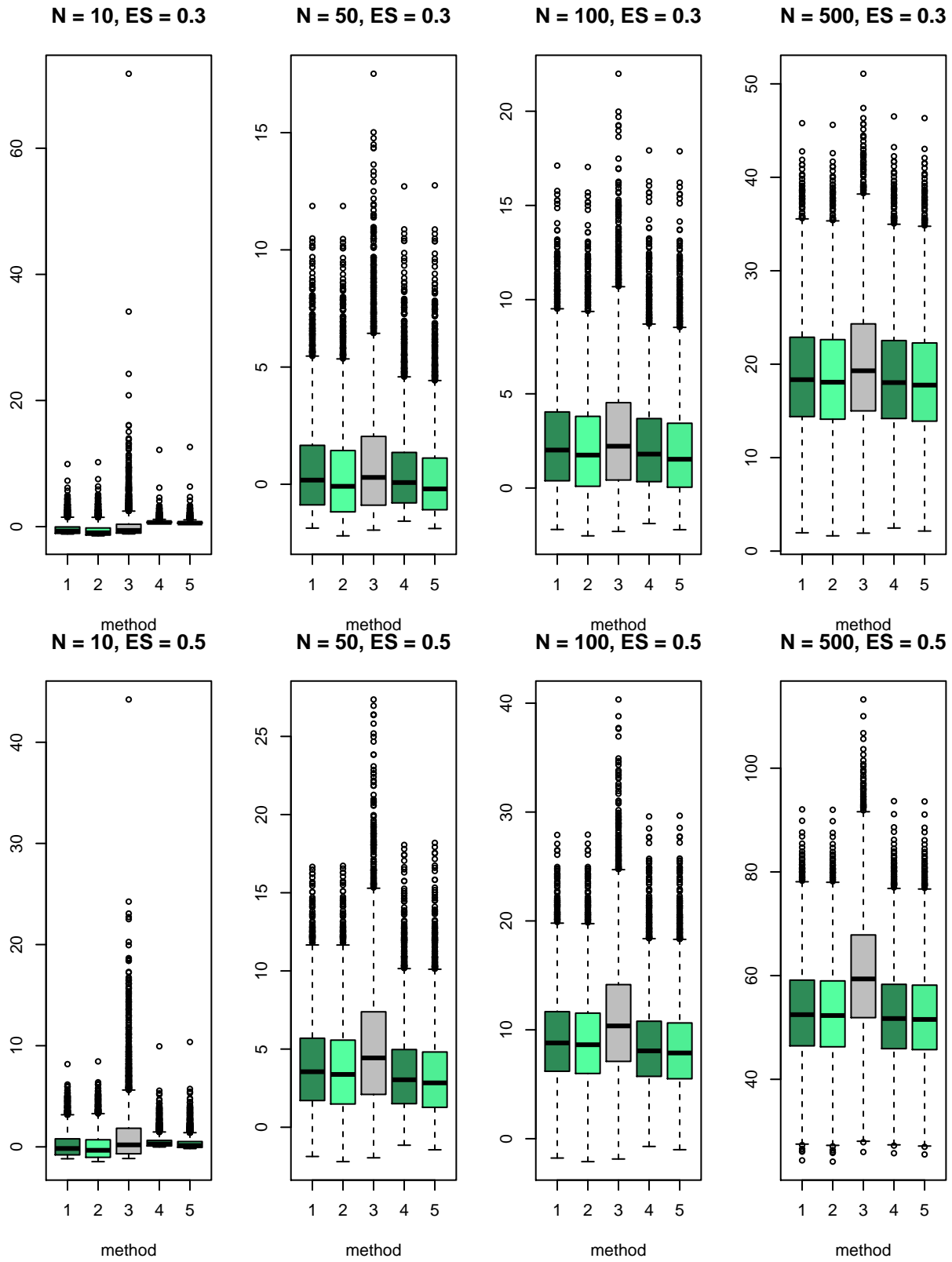
[illegible]

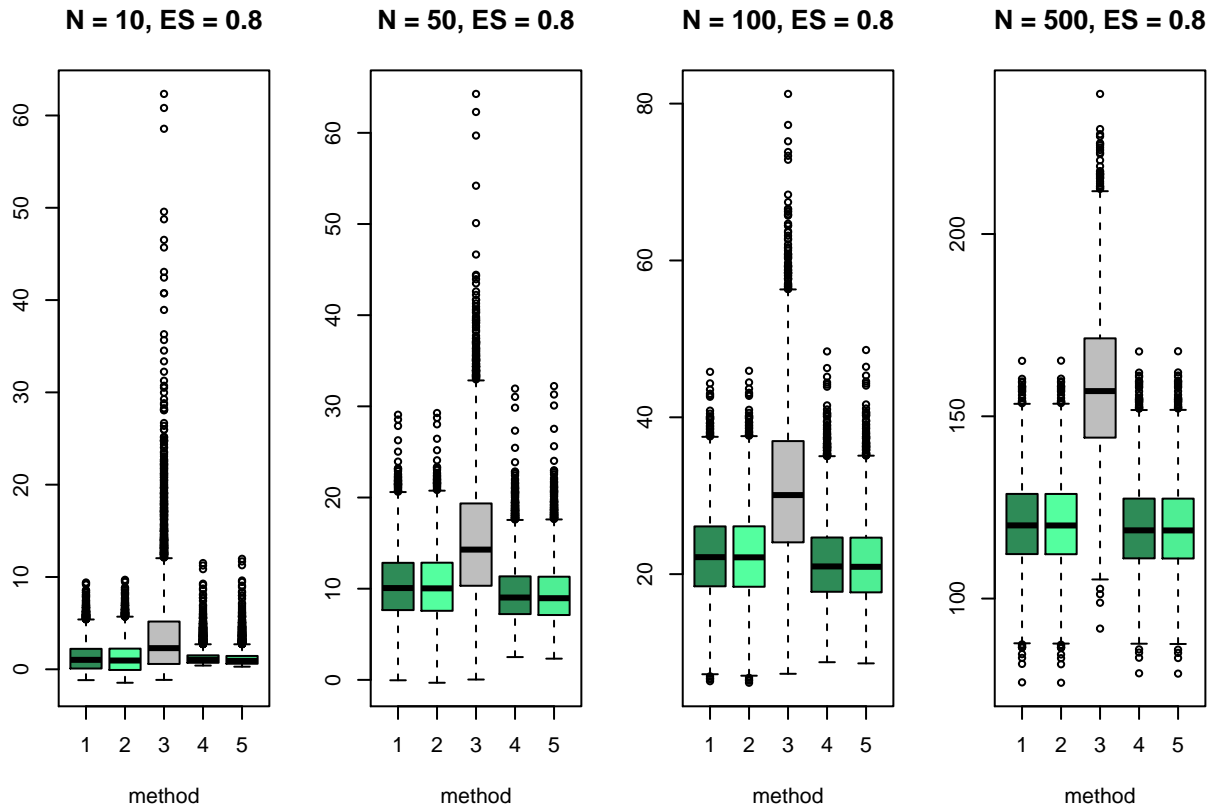
Colors:

- ```
bxpdat <- list()
par(mfrow=c(1,4), mar=c(5, 3, 4, 1))

for(i in 1:(length(Ns)*length(deltas))){
 bxpdat[[i]] <- as.data.frame(matrix(NA, nrow=5*5000, ncol=2, dimnames = list(NULL, c("method", "BF"))
 bxpdat[[i]]$method <- rep(1:5, each=5000)
 for(j in 1:5){
 bxpdat[[i]]$BF[bxpdat[[i]]$method==j] <- res[, i, j]
 }
 boxplot(BF ~ method, bxpdat[[i]], main=paste0("N = ", conds[i, 1], ", ES = ", conds[i, 2]), col=c("se", "se", "se", "se", "se"))
}
```







#### Observations:

- Across all sample sizes and effect sizes, the parabola provides a better approximation to the Bayes factor distribution than Jeffreys' approximation
- Jeffreys's approximation always provides an overestimate
- With different N's and effect sizes, it also becomes clear that Jeffreys's approximation suffers from not being able to adjust (properly) to the prior
- The constant  $A$  could be used for further flexibility, but as I showed before, it does not improve the estimation for individual Bayes factors